Fusion Plasma Reconstruction

Google Applied Sciences
Presented by Ian Langmore
At the UQ Summer School - 2019 - USC

TAE and the Plasma Debugger

Please interrupt me and ask questions!

Google -- TAE Partnership

Goal:

Accelerate development of viable fusion energy

tae TECHNOLOGIES

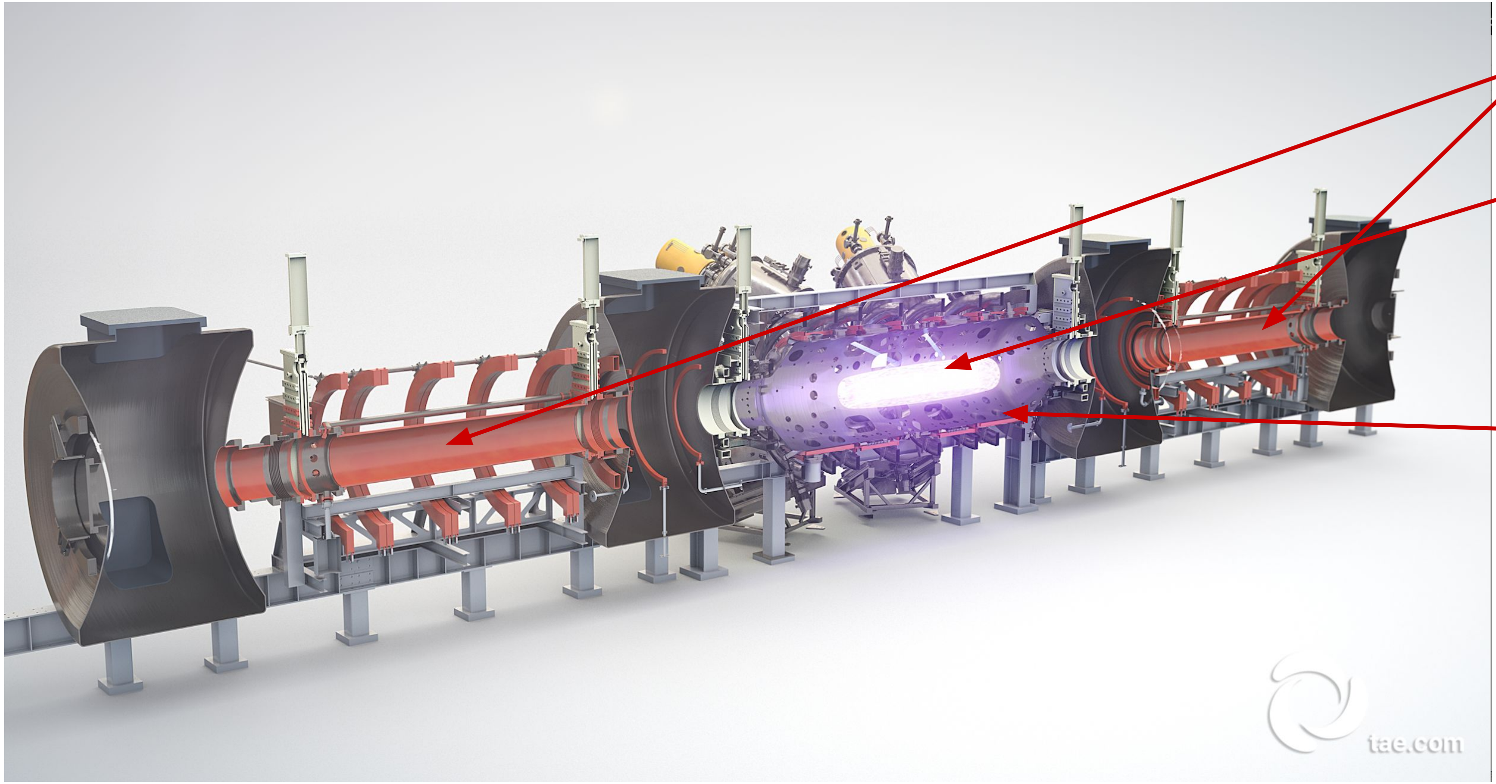
- Commercial fusion energy company
- Southern California
- TAE personnel on project
  - M. Binderbauer, D. Ewing, A. Smirnov
  - E. Trask, H. Gota, R. Mendoza, J. Romero, S. Dettrick

Google!

Google (Applied Sciences)

- Commercial web-search company
- Northern California
- Google personale on project (order of joining)
  - R. Koningstein, J. Platt
  - T. Baltz, M. Dikovsky, I. Langmore, T. Madams, P. Norgaard, Y. Carmon, N. Neibauer, R. von Behren

Norman: Experimental FRC Plasma Generator



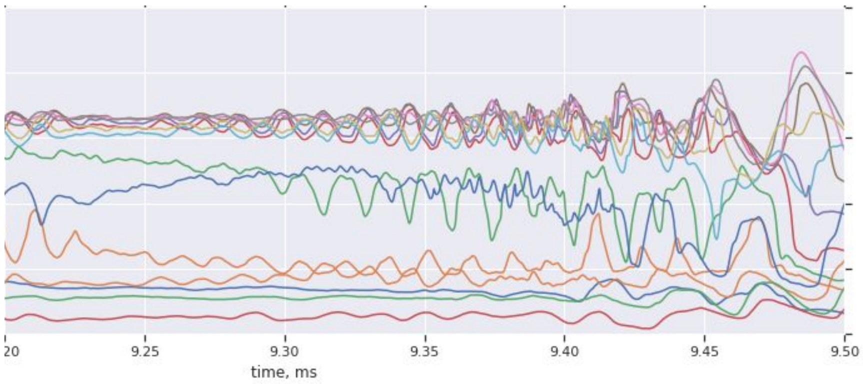
Plasma formed on each end, then fired into center vessel

Plasma confined by magnetic fields, heated/stabilized by neutral beams

Ports provide access for measurement devices

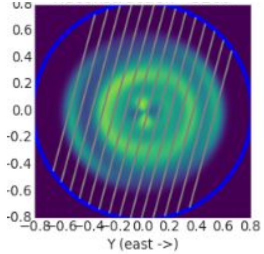
Goal: Learn to confine plasma long enough, at high enough temperatures, en route to net positive energy (in later machine)

Measurements in → Reconstructed plasma out

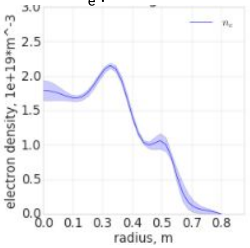


Interferometer traces

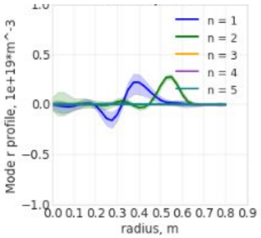
Cross section



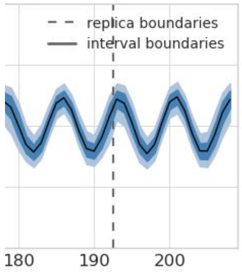
N_e profile



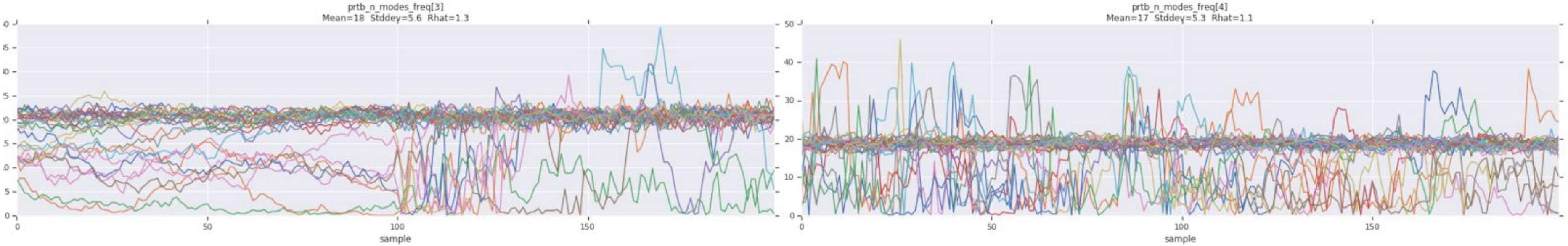
Mode profiles



Mode N=3 vs time

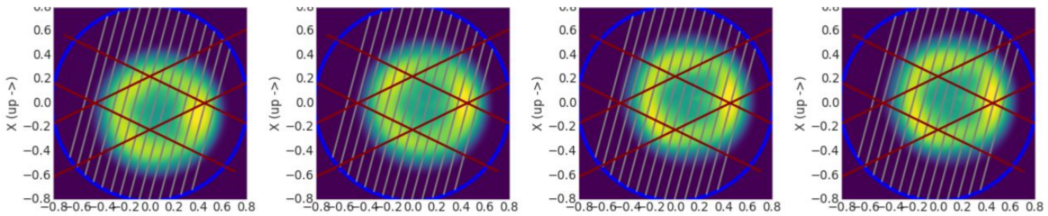


Reconstructions are Samples of Plasmas



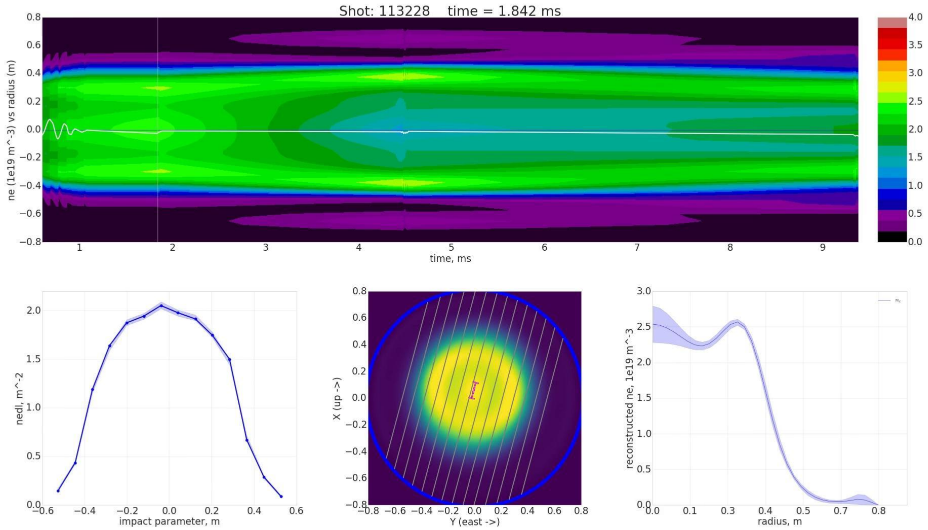
Samples of random variables

Map to samples of plasmas



Resolving Plasma Properties: The Center

Location parallel to lasers is *not well resolved* by Interferometer alone



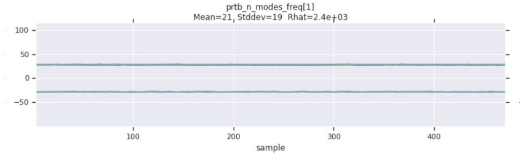
Coupled SEE helps to resolve this

Blue dots are samples from posterior over plasma center

Resolving Plasma Properties: Mode Rotation

Rotation direction is *not well resolved* by Interferometer alone

Coupled magnetic probes help to resolve this



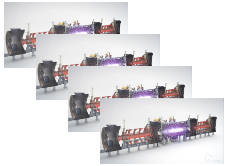
Bimodal: ± 28 kHz

Traces from
Markov Chain

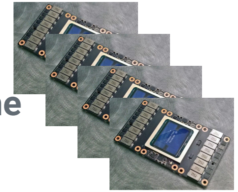
Unimodal: -28 kHz

Computation is *highly* parallelized

1000+ experiments



5000+ GPUs located around the world



On each GPU

Plasma density etc… represented by <u>many</u> 4-D Tensors, each of shape

`[n_samples, n_chains, n_times, n_events]`

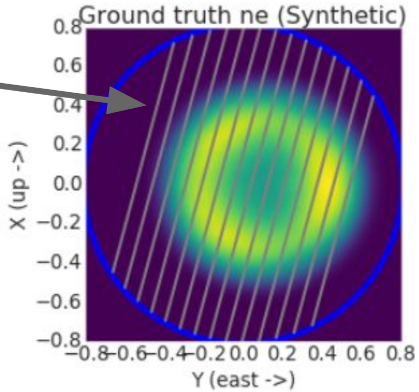
- `n_samples`: Number of samples from each chain
- `n_chains`: Number of Markov chains run in parallel
- `n_times`: Number of times handled by each GPU
- `n_events`: Dimension of each random variable

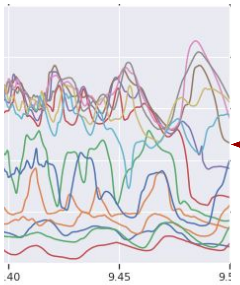
Example shape: `[450, 30, 20, 16]`

Some Bayesian Modeling Details

Modeling the Interferometer

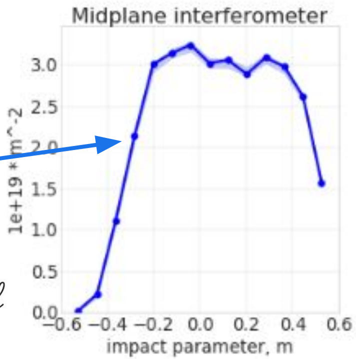
14 Lasers pass through plasma





Measure phase shift

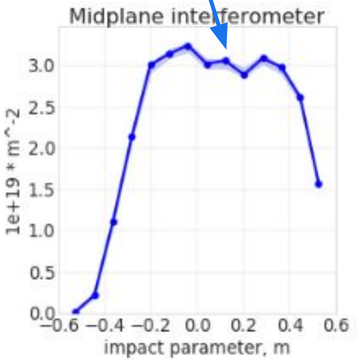
Phase shift translated to integrated density



$$\Delta\text{Phase}_i = C \int_{-1}^{1} N_e(x_i + \ell v)\, d\ell$$

Interferometer Forward Model

Phase shift translated to integrated density



Ideally, with N_e^{true} the *actual* plasma density,

$$\Delta\text{Phase}_i = C \int_{-1}^{1} N_e^{true}(x_i + \ell v)d\ell$$

We model density as $N_e = N_e(\xi)$, for $\xi \sim \mathcal{N}(0, I)$.

N_e is defined on a discrete grid.

Our *Forward Model* for (phase) measurement $m = (m_1, \ldots, m_{14})$ is

$$m = AN_e + \sqrt{\sigma_{const}^2 + \sigma_{prop}^2 AN_e} \cdot \epsilon, \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, I)$$

$$(AN_e)_i \approx C \int_{-1}^{1} N_e(x_i + \ell v)d\ell$$

Most sources of “noise” can be *modeled* as random variables

Model for Electron Density (N_e)

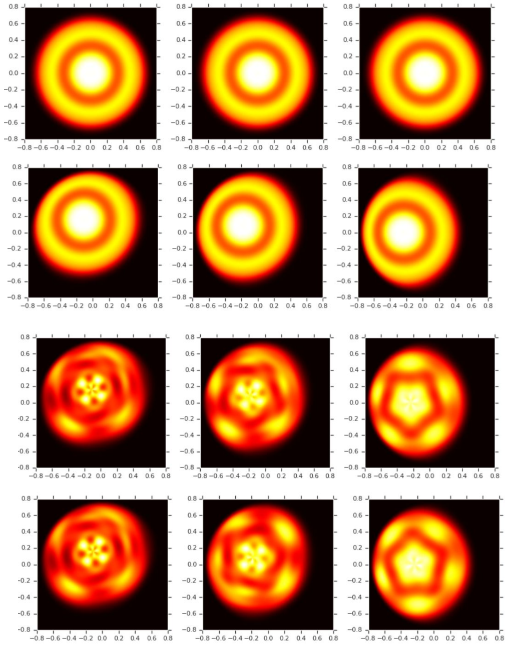
Our Prior is over these

$$N_e(r,\theta) = \log\left[1+\exp\left\{\sum_{k=1}^{K}\xi_k u_k(r)\right\}\right], \quad \text{where} \quad \sum_{k=1}^{\infty} u_k(r)u_k(r') \to \exp\left\{-\frac{|r-r'|^2}{2(0.15)^2}\right\}, \quad \text{and} \quad \xi_k \sim \mathcal{N}(0,(0.1)^2)$$

$$(r\cos\theta, r\sin\theta) \mapsto (r\cos\theta-\delta_x, r\sin\theta-\delta_y), \quad \text{where} \quad \delta_x,\delta_y \sim \mathcal{N}(0,(0.1)^2)$$

$$N_e(r,\theta) \mapsto N_e(r,\theta)\left[1+\mathrm{Bound}_{(-1,1)}\left(\sum_{n=1}^{N}\eta_n\sin(n\theta)\right)\right], \quad \text{where} \quad \eta_n \sim \mathcal{N}(0,1/n^2).$$

Now turn every random variable into a random process in time...



Posterior: Putting the model together

$$\text{Posterior} \propto \text{Prior x Likelihood}$$
$$p(z \mid m) \propto p(z)p(m \mid z)$$

Prior: Weak physical constraints

Fun to think about introducing more physics

...but physicists would rather know what the measurements are saying *independent of any model*

Likelihood: Super accurate instrument model

Bayesian Inverse Problems

⇒ ~~Just another Generative Model~~

⇒ Solving equations (with random coefficients)

...If you have the wrong equations, you'll get the wrong solution

Inference: The MAP Estimate

$$Z_{MAP} := \arg\max_{z} p(z)p(m \mid z) = \arg\max_{z} p(z \mid m)$$

MAP estimates:

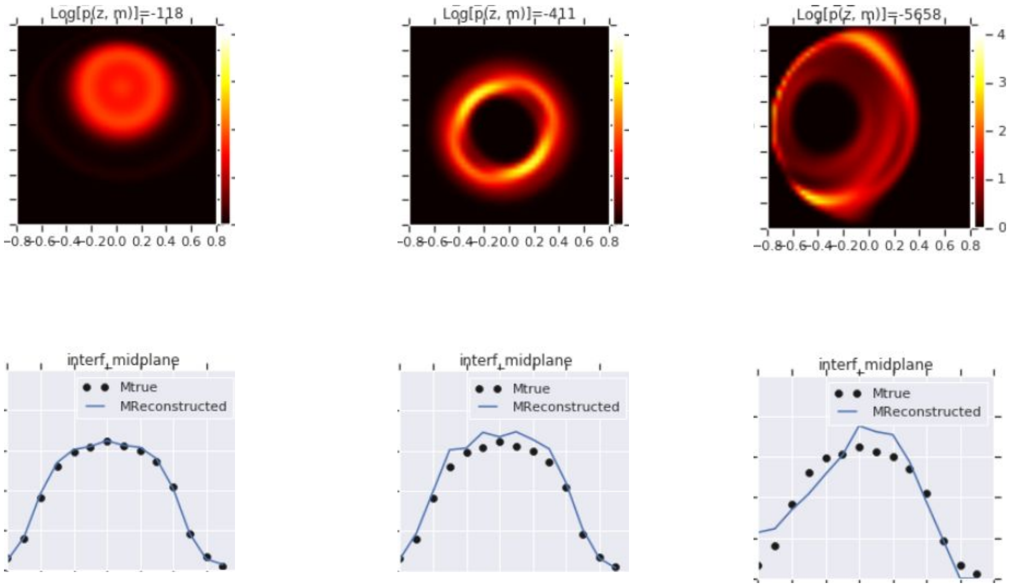
Attempt at a “best guess”

Warning:

Often finds “bad random modes”

⇒ Compute 30 estimates in parallel

(good use of TFP batch dimension capabilities)



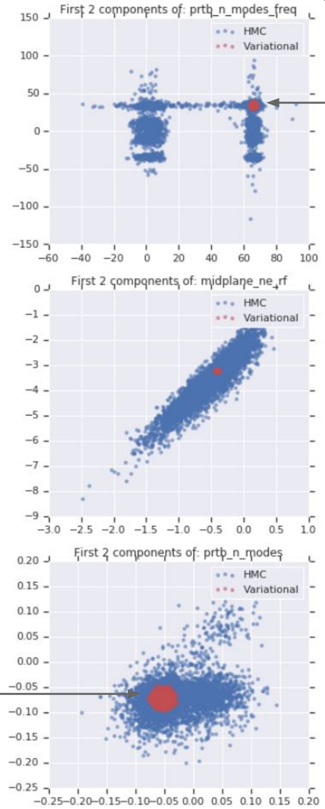
Variational Inference : Could not make it work

Start with parameterized distribution $q(z;\phi)$. Then set

$$\phi^* := \arg\min_{\phi} \int \log\left[\frac{q(z;\phi)}{p(z,m)}\right] q(z;\phi)dz$$
$$= \arg\min_{\phi} KL\left[q(z;\phi)||p(z\,|\,m)\right].$$

- $KL[q||p] = 0 \Leftrightarrow q = p$
- Above loss function has a stable numerical approximation

- In most problems, there is no ϕ such that $q = p$
- Often under-estimates uncertainty



Random mode found ;)

VI is “scared” of putting mass outside the extent of p(z)

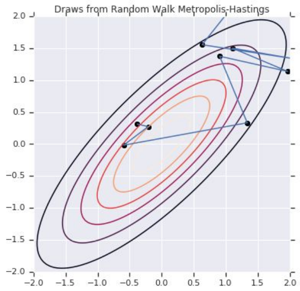
Every time a compromise must be made, q(z) will error in this manner.

Sampling: Random-Walk Metropolis Hastings

Metropolis Hastings recipe to sample from $p(z)$

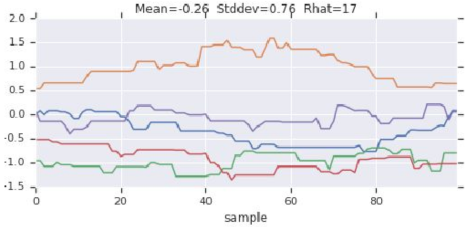
1. Initialize $z = z^0$
2. Propose a move $z \to y \sim q(y|z)$
3. Accept with probability $\min\left\{1, \frac{q(z|y)p(y)}{q(y|z)p(z)}\right\}$
    1. If Accept, set $z^1 = y$
    2. If Reject, set $z^1 = z^0$
4. Iterate...

Random Walk Metropolis-Hastings if $q(y|z) \sim \mathcal{N}(y; z, \sigma^2 I)$ is Gaussian



Random Walk behavior $\Rightarrow$ slowly mixing chains in higher dimensions

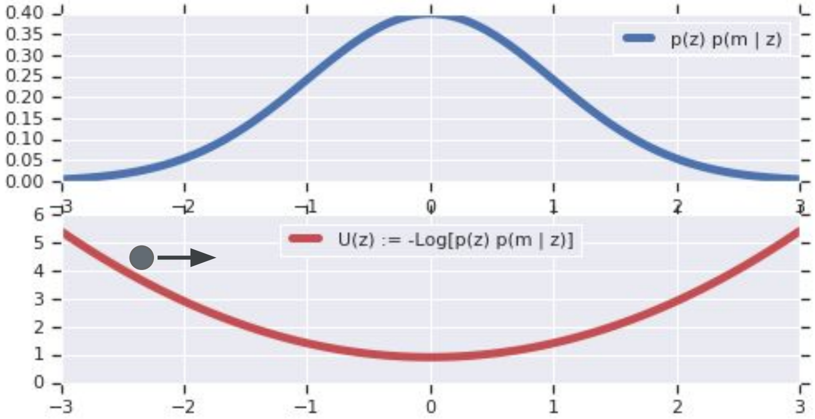
5 chains sampling a 50-dimensional Gaussian:
Pictured is the first component



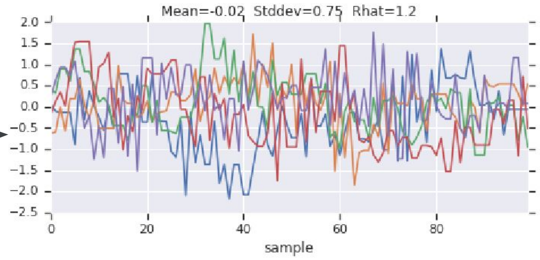
HMC: A proposal that scales well

Physics explanation

1. Let $U(z)$:= -Log[p(z) p(m | z)] define a surface (as a function of z in R^N)
2. Start a ball at $(z^0, U(z^0))$
3. Give the ball a random “kick”
4. Let the ball roll for time T, giving you the proposal



If well tuned, the “rolling” allows the proposal to travel a long distance



Neal. 2012

HMC: A proposal that scales well

Increase dimension $\mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ by adding "momentum" ζ, and then...

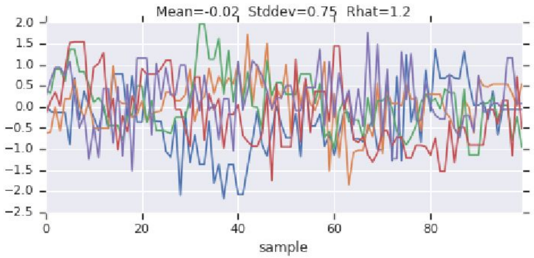
1. Initialize $(z, \zeta) = (z^0, \zeta^0)$, where $\zeta^0 \sim \mathcal{N}(0, I)$
2. Define $H(z, \zeta) := -\log p(z) + \|\zeta\|^2/2$
3. Propose $(z(T), \zeta(T))$, the time-T (numerical) solution to the initial value problem:

$$\dot{z}(t) = \frac{\partial H}{\partial \zeta}, \quad z(0) = z^0,$$
$$\dot{\zeta}(t) = -\frac{\partial H}{\partial z}, \quad \zeta(0) = \zeta^0,$$

4. Accept with probability

$$\min\left\{1, \exp\{H(z^0, \zeta^0) - H(z(T), \zeta(T))\}\right\}$$

If numerical integration was perfect, you would accept *every time*



This produces samples $[(z^0, \zeta^0), \ldots, (z^K, \zeta^k)]$ from $p(z, \zeta) \propto \exp\{-H(z, \zeta)\}$

The samples $(\zeta^0, \ldots, \zeta^K)$ may be discarded.

The samples $(z^0, \ldots, z^K)$ are from $p(z)$.

Neal. 2012

HMC Efficiency Tradeoff

Smaller numerical integration step size ⇒

- Lower integration error
- Higher Prob[Accept]

But also…

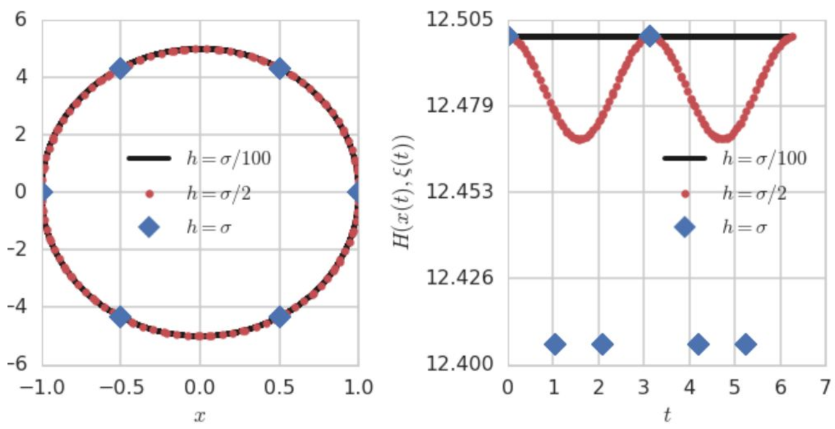
- number of steps needed

  ~ O(1 / step_size)

Asymptotically, optimal step_size gives

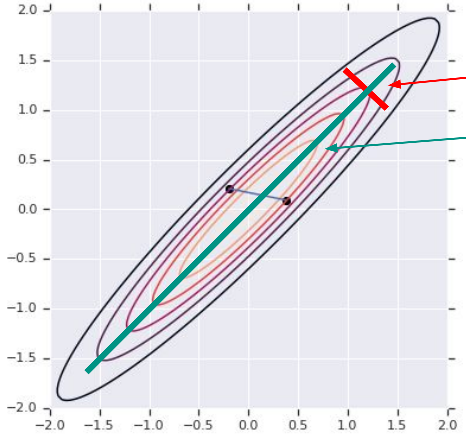
P[Accept] ≈ 0.68

Beskos. 2010



Integration error due to finite step size

Influence of Geometry



What are the optimal step size h^* and number of integration steps ℓ^*?

- h must be small enough so integration can navigate smallest scales
- $T = h\ell$ must be large enough to traverse largest scales

In the Gaussian case, with $\mathrm{Eig}(\mathrm{Covariance}) = \sigma_1^2 \geq \cdots \geq \sigma_n^2$:

$$h^* \propto \left(\sum_{i=1}^{n} \frac{1}{\sigma_i^4} \right)^{-1/4}, \qquad \ell^* \propto \kappa := \left(\sum_{i=1}^{n} \frac{\sigma_1^4}{\sigma_i^4} \right)^{1/4}$$

L. 2019

Linear Preconditioning

Suppose

$$\mathrm{Cov}(Z) = E[ZZ^T] = C = LL^T$$

Rather than sampling from Z, sample from

$$\tilde{Z} := L^{-1}Z,$$

and then since $\mathrm{Cov}(\tilde{Z}) = I$

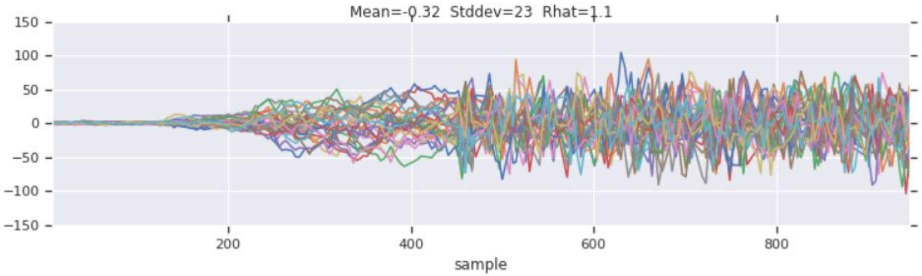
$\kappa(\tilde{Z}) = n^{1/4}$, and $\tilde{Z}$ is perfectly conditioned

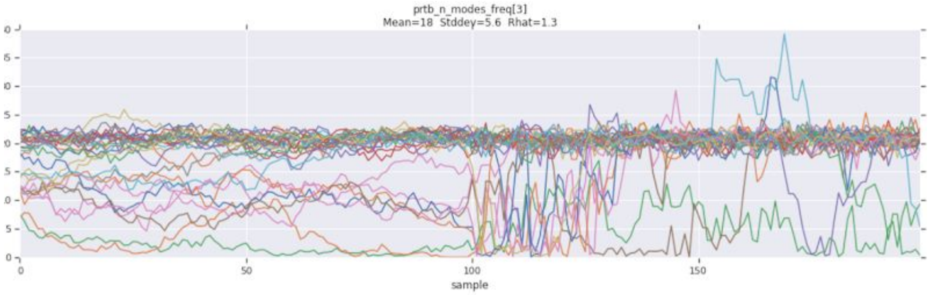
Practicalities

- You don't know C, so you must estimate it using...
  - samples
  - variational inference
  - `tf.hessians`
- Have to hope nonlinearities don't mess things up!

Sampling Strategy : Iterative improvement

1. Find preconditioner L via Variational Inference
2. Use `tfp.mcmc.SimpleStepSizeAdaptation` to adapt h until P[Accept] ≈ 0.9
3. Draw ~ 25 samples from 30 parallel chains
4. Update L → Diag(Stddev(Z_{sample}))
   a. adapt step size again
5. Draw ~ 25 more samples
6. Update L → ?? Depending on estimated change in Kappa
   a. adapt step size again
7. Continue, until Rhat is small enough





Set L → Diag(Stddev)

Evaluation of Bayesian Reconstructions

First: Let’s be realistic

Do we really sample from the “posterior”?

- Our prior is “reasonable”, but is it really the marginal distribution over all possible plasmas?
  - hahahahhahahaha
- We model many effects, but plasmas are complex beasts and we do not model all
- We only have one measurement, of much smaller dimension than our unknowns.
- We *never* sample from the tails
  - takes too long to get samples
  - *by definition* you can’t really validate them
- Will we ever know we’re right about anything?
  - we have *zero* golden data

Responsible hypothesis generation

Debugger Commandments:

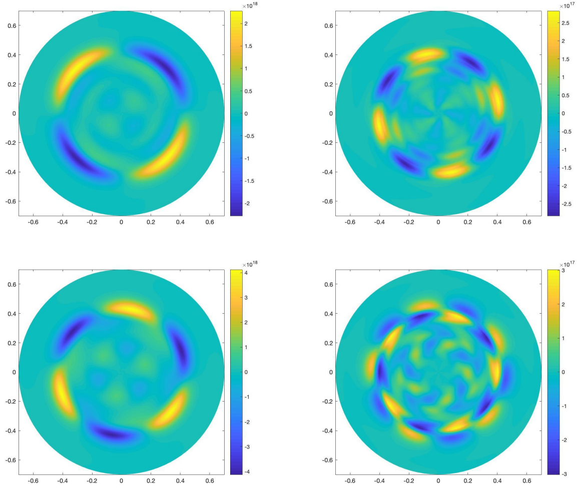
1. If a ~~human~~ physicist can infer interesting event X is likely from the raw data, so too shall the debugger
2. If two events, X and Y are both somewhat likely, the debugger shall indicate thus
3. The debugger shalt not send TAE on too many wild goose chases for effects it has *hallucinated*
4. The degree to which we achieve 1-3 shalt be exhaustively tested using synthetic data

Synthetic Plasma

No "ground truth" solution exists for plasma dynamics (can't solve for 10^{20} particles + Maxwell's equations).

Approximate solutions from fluid/particle simulation can still be used to test the inference algorithm.
A physicist combines and modifies certain features from simulation data to make a "synthetic plasma".

Examples:
- Check for false positive / false negative of feature identification
- Evaluate impact of 3d effects on 2d reconstruction
- Investigate cases with statistical ambiguity



"What is the smallest density fluctuation that can be reconstructed?"

"Can the model identify both fast and slow feature dynamics?

"How does aliasing present with high-frequency behaviors?"

MCMC Diagnostic : Rhat

Running parallel Markov Chains...

- makes efficient use of GPUs
- allows for the convergence diagnostic R-hat
    - tfp.mcmc.potential_scale_reduction

$$\hat{R} := \frac{\text{WithinChainVariance} + \text{BetweenChainVariance}}{\text{WithinChainVariance}}$$
$$= \frac{N_c^{-1} \sum_{n=1}^{N_c} \sigma_n^2 + (N_c - 1)^{-1} \sum_{n=1}^{N_c} (\mu_n - \mu)^2}{N_c^{-1} \sum_{n=1}^{N_c} \sigma_n^2}$$

If chains are mixing well

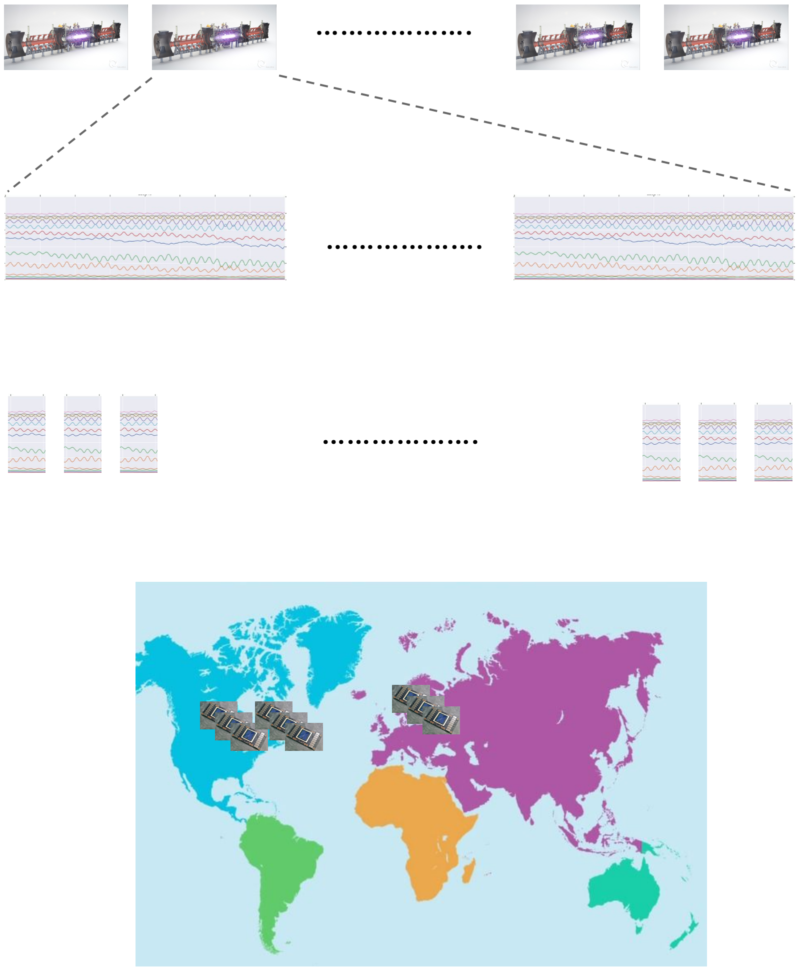
$$\hat{R} \to 1, \quad \text{as} \quad N_c \to \infty.$$

Generally, $\hat{R} < 1.1$ is "good enough" for us.

Bayesian Inverse Problems at Scale

One does not simply...

...divide 50 petaflops of compute across 15,000 time points from 1000+ experiments



1000+ experiments

15k time pts per experiment

Divide each experiment into *bursts* of length 20

Distribute bursts across multiple data centers

Resource Sharing : Problems and Solutions

- We want access to 5000+ GPUs ⇒ Google has them
- Other teams need access ⇒ Borg cluster management helps share
- Some jobs are more important than others ⇒ High priority jobs can *preempt* lower priority ones
- We are not the only important team at Google ⇒ Must checkpoint results / recover from checkpoints
- Jobs from the same experiment finish at different times ⇒ Process asynchronously in a queue

Verma. 2015

Computational Workhorse: GPUs

Why GPUs?

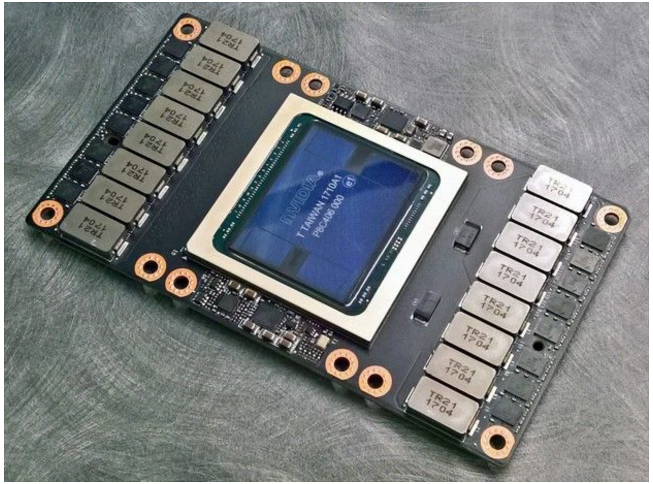
- Performance scales well as arrays get larger
  - Prefers doing a *small number* of *large* (e.g. batch) operations (e.g. MatMul)

How?

- TensorFlow [Probability] compile to CPU or GPU

How many?

- Typically using 5000+ GPUs at any given time



Scaling up: Number of people

Cannot underestimate the importance of this...

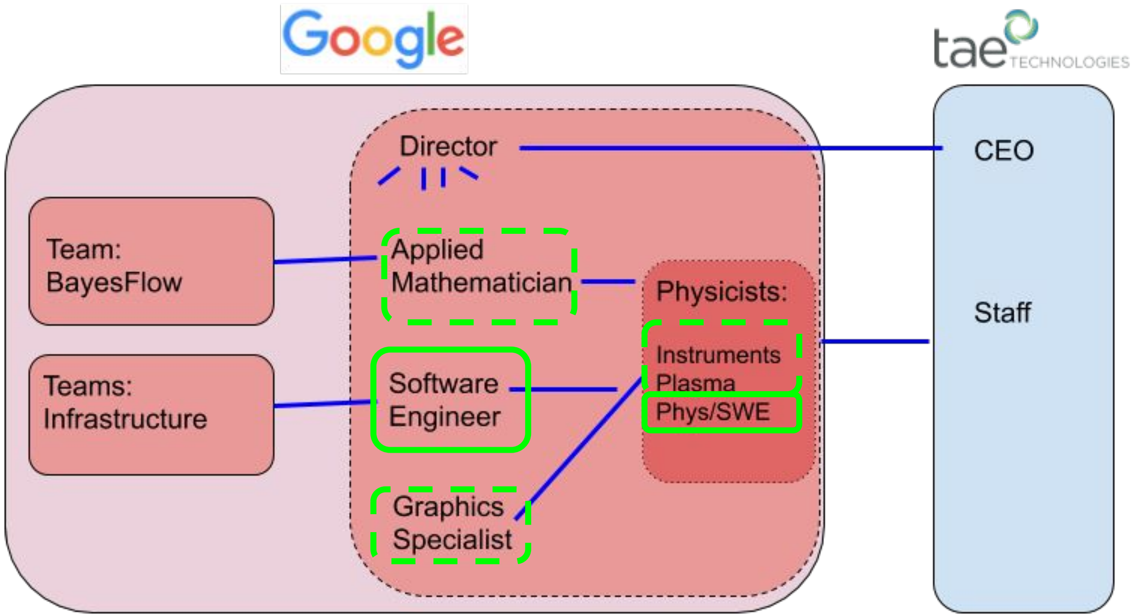
How to keep people happy and productive?

How to get the most from every team member?

Strategy:

Let everyone be the expert/boss of their own domain

People are Nodes in a Collaborative Network

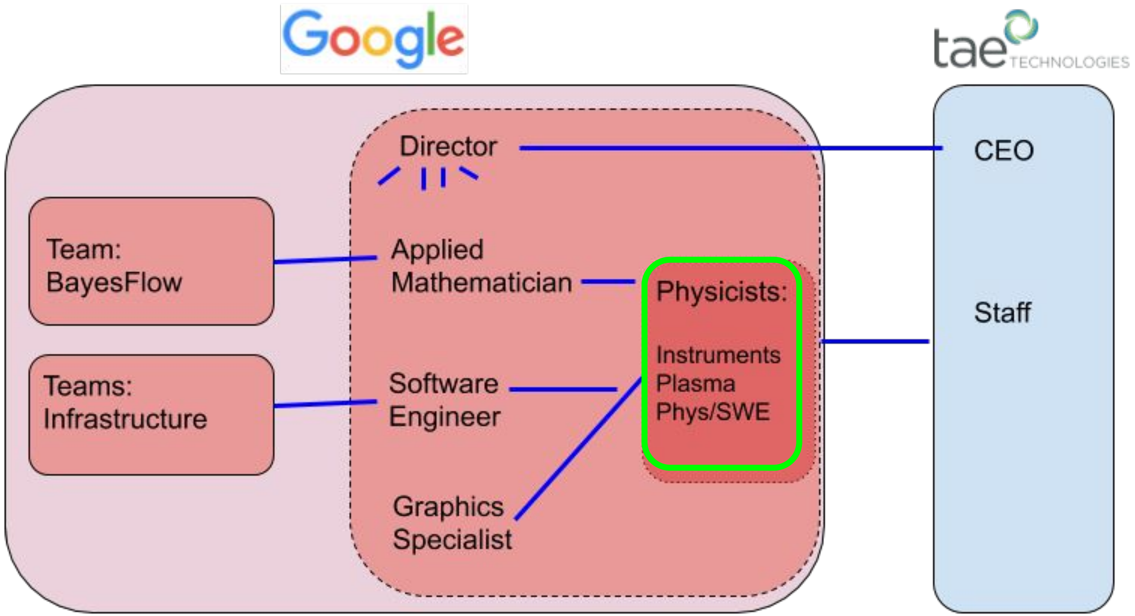


Software Engineer (SWE) Tasks

- Set up queueing system to run jobs
- Automating large-scale evaluation/debugging/ reconstructions
- Resource management

… everyone is a bit of software engineer -- writes ~ 300 lines of code a day

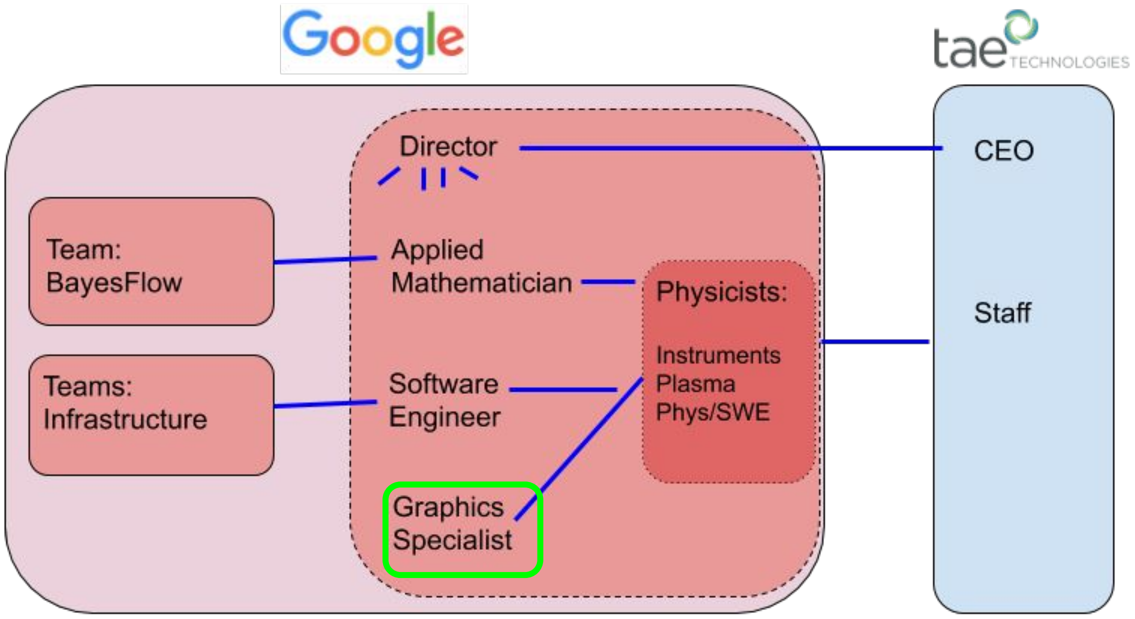
People are Nodes in a Collaborative Network



Physicist tasks

- Prepare synthetic/simulated plasmas and measurements
- Model instrumentation
- Model priors
- Coordinate with TAE

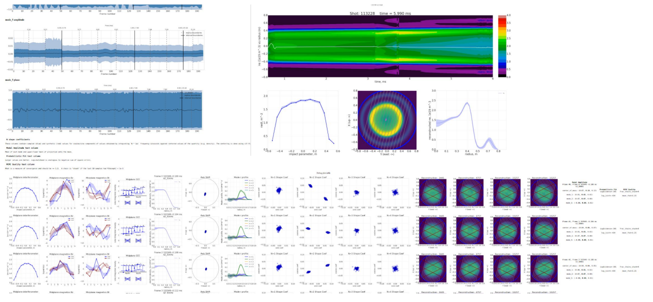
People are Nodes in a Collaborative Network



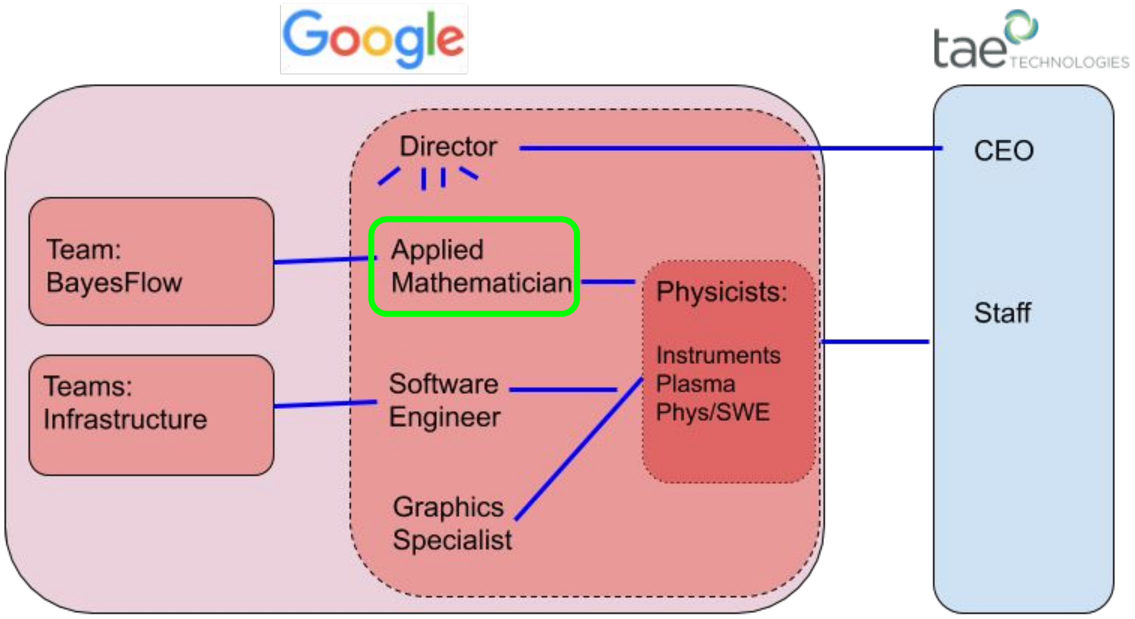
Graphics specialist tasks

- Maintain and grow *huge* collection of automatically generated images/videos

  For TAE's analysis -- For our debugging



People are Nodes in a Collaborative Network

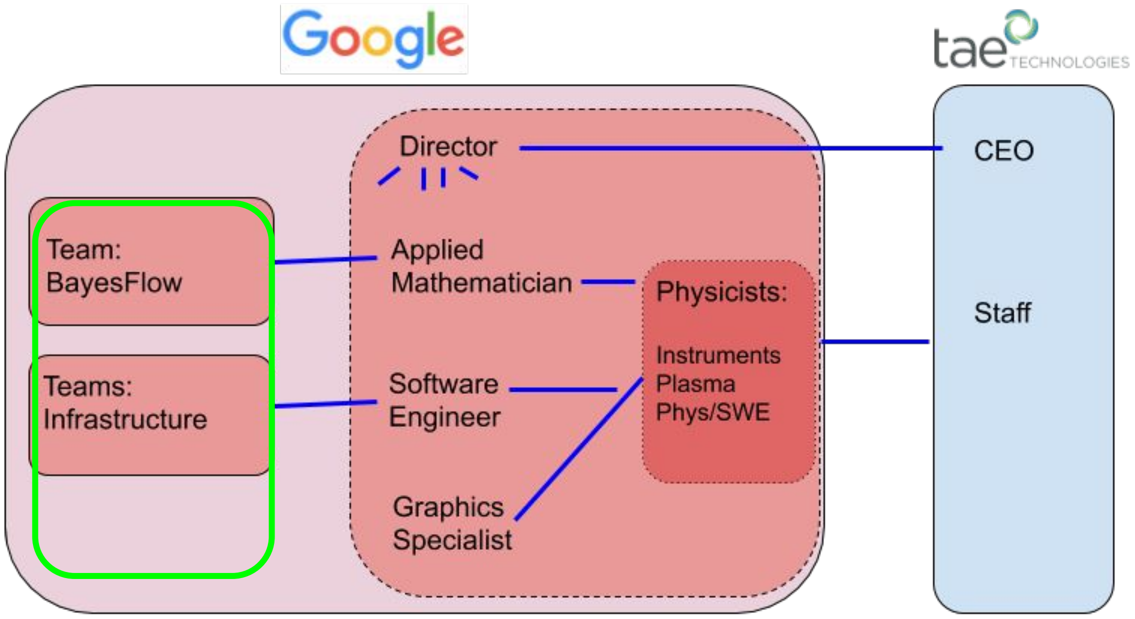


Applied Mathematician Tasks

- Work with BayesFlow team to write core statistical tools
- Write debugger code that translates our problem into a Bayesian context
- Communicate/Teach Bayesian concepts to rest of team

Colab (IPython notebook) very useful for this

People are Nodes in a Collaborative Network



The rest of Google

- Probabilistic code (TensorFlow Probability) developed by BayesFlow
- TensorFlow team
- Cluster/Infrastructure teams
- Source code management
- Q&A forums
- Food services
- etc...

Thank You!

Contact: langmore@google.com

Beskos et al. 2010: Optimal tuning of the Hybrid Monte-Carlo algorithm (link)

Betancourt. 2018: A conceptual introduction to Hamiltonian Monte Carlo (link)

Hoffman et al. 2019: NeuTra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport (link)

Langmore. 2019: A condition number for Hamiltonian Monte Carlo (link)

Neal. 2012: MCMC Using Hamiltonian dynamics (link)

Parno, Marzouk. 2017: Transport map accelerated Markov chain Monte Carlo (link)

Verma et al. 2015: Large scale cluster management at Google with Borg (link)

Citations

Supplementary Material

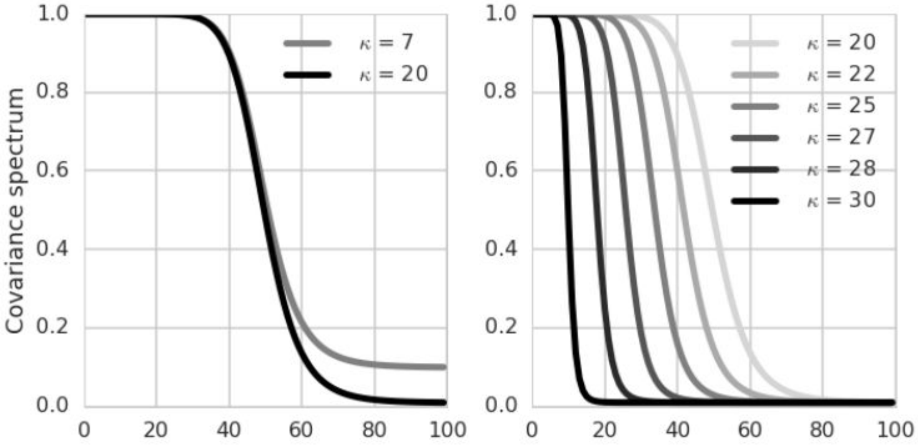
Preconditioning : Heuristics

Recall the (Gaussian) result on number of integration steps required for efficient sampling:

$$\ell^* \propto \kappa := \left(\sum_{i=1}^{n} \frac{\sigma_1^4}{\sigma_i^4} \right)^{1/4}$$

Minimized when all eigenvalues are the same

Worst if one large eigenvalue and many small



Bounds and Asymptotic Results for HMC

People mostly use the (second order) "leapfrog" integrator, so, with step size h,

$$\left|H(Z(T),\zeta(T)) - H(Z^0,\zeta^0)\right| \leq CTh^2.$$

In distribution, the (asymptotic) bound is much better

$$H(Z(T),\zeta(T)) - H(Z^0,\zeta^0) \sim \mathcal{N}(\alpha h^4/2, \alpha h^4)$$

This implies (after some work) that for maximal efficiency

$$P[Accept] \approx 0.68$$

Beskos. 2010

In the case $p \sim \mathcal{N}(\mu, C)$, the h that achieves this is

$$h^* \propto \left(\sum_{i=1}^{n} \frac{1}{\sigma_i^4}\right)^{-1/4},$$

where $\sigma_1^2 \geq \cdots \geq \sigma_n^2$ are the eigenvalues of C.

Since we should set the integration time $T \propto \sigma_1$, and also $T = h\ell$, where ℓ is the number of integration steps, we have

$$\ell^* \propto \kappa := \left(\sum_{i=1}^{n} \frac{\sigma_1^4}{\sigma_i^4}\right)^{1/4}$$

L. 2019

Actually...

complex prior transformations are done inside the Forward Model

Our prior (previous slide) a pushforward of a Gaussian

With $\phi \sim \mathcal{N}(0, I)$

$$p(z) = G_{\#}\phi(z) = \phi(G^{-1}(z))|DG^{-1}(z)|$$

and we actually sample from this pushforward:

$$\begin{aligned}\tilde{p}(z) :&= G_{\#}^{-1} p(z \mid m) \propto G_{\#}^{-1} p(z) p(m \mid z) \\ &= \phi(z) p(m \mid G(z))\end{aligned}$$

then transform back to get our final samples

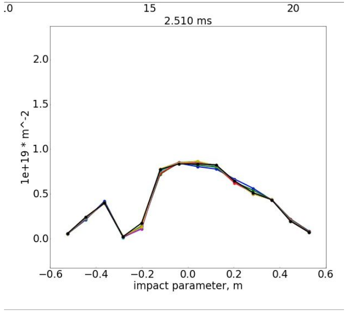
$$Z = G(\tilde{Z}), \quad \text{where} \quad \tilde{Z} \sim \tilde{p}$$

Why do this?

- Does *not* change the model
- Computational savings: No need to evaluate G^{-1}
- Same steps work when G is not invertible
    - e.g. G involves absolute values

Interferometer Fringe Jumps

Phase is lost ⇒ Integrated density is *wrong*



Scaling up Inference and Evaluation

Goals

For about 1000 experiments…

- generate 500 *effective* samples (of plasmas)

  ...at 15,000 time points

  ...in less than one day

Use these reconstructions…

- to understand exactly what happened during important experiments
- to recognize patterns across experiments